VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Tvorba internetového systému pro prezentaci zájmových bodů - část mobilního rozhraní

Internet System for Presentation of Points of Interest - part mobile interface

2019 Matěj Nedojedlý

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Matěj Nedojedlý**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Tvorba internetového systému pro prezentaci zájmových bodů - část mobilního rozhraní
Internet System for Presentation of Points of Interest

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit platformně nezávislou serverovou aplikaci a aplikaci pro systém Android pro prezentaci zájmových bodů a navigaci k nim v rámci obcí, měst či rozsáhlejších podniků.

1. Proveďte rešerši možných řešení serverové části včetně databáze a komunikačního protokolu s databází.
2. Realizujte a otestujte serverovou část.
3. Navrhněte aplikaci na platformě Android pro komunikaci s databází.
4. Realizujte a otestujte aplikaci na platformě Android (v součinnosti s vedoucím práce).

Seznam doporučené odborné literatury:

[1] POKORNÝ, Jaroslav a Michal VALENTA. *Databázové systémy: vysokoškolská učebnice.* Praha: České vysoké učení technické v Praze, 2013. ISBN 978-80-01-05212-9.

[2] LACKO, Ľuboslav. *Vývoj aplikací pro Android.* Přeložil Martin HERODEK. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

[3] HELLMAN, Erik. *Android programming: pushing the limits.* Chichester: Wiley, 2014. ISBN 978-1-118-71737-0.

[4] CASTLEDINE, Earle, Myles EFTOS a Max WHEELER. *Vytváříme mobilní web a aplikace pro chytré telefony a tablety.* Přeložil Jakub MUŽÍK. Brno: Computer Press, 2013. ISBN 978-80-251-3763-5.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Skapa, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

prof. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2019 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 30. dubna 2019 .....

Rád bych poděkoval vedoucímu práce panu ing. Janu Skapovi, Ph. D. za vedení práce.

Abstrakt

Cílem bakalářské práce je vytvořit mobilní aplikaci pro operační systém Android, pro prezentaci zájmových bodů v okolí zařízení a pro vybrané společnosti. Práce zahrnuje implementaci vlastní databáze a PHP souborů, pomocí které s ní aplikace komunikuje. Aplikace využívá Google API pro zobrazení map, k využívání dat z Google databáze a tvoření tras mezi body. Aplikace zahrnuje registrování nových uživatelů, přidávání nových a následné upravování zájmových bodů v databázi.

Klíčová slova: Android, Google API, PHP

Abstract

The aim of the bachelor thesis is to create a mobile application for the Android operation system to present points of interest in the range of device and for selected companies. The work includes implementation of own database and PHP files which the application communicates with it. The application uses Google's API to view maps, use data from Google database and create routes between points. The application includes registration for new users, adding new ones and then editing points of interests in database.

Key Words: Android, Google API, PHP

Obsah

Seznam použitých zkratek a symbolů

XML	–	Extensible Markup Language
API	–	Application Programming Interface
PHP	–	Hypertext Preprocessor
HTTP	–	Hypertext Transfer Protocol
URL	–	Uniform Resource Locator
JSON	–	JavaScript Object Notation
JIT	–	Just in time

Seznam obrázků

Seznam výpisů zdrojového kódu

Úvod

Určitě se vám někdy stalo, že jste nemohli najít místnost v nové práci, či učebnu ve škole. Přiložené mapy nebyly kompatibilní s vaším mobilním telefonem, nebo místa nebyla přímo vyznačena. Tento problém by mohla vyřešit má aplikace pro platformu Android. Aplikace má pomoci nově příchozím studentům této vysoké školy v orientaci na kampusu.
Aplikace je napsána v programu Android studio a hlavním programovacím jazykem je Java. Dalším použitým jazykem je XML, pomocí kterého jsem realizoval grafickou část aplikace. Pro implementaci map jsem využil knihovny od GoogleAPIs, o kterých se rozepíši v následujících kapitolách. Aplikace umí pracovat s vlastní databází. Databáze dokáže uchovat informace o uživatelích, firmách, místech dané společnosti a vytvořených tras. Dané místa lze pomocí vyhledávání zobrazit na mapě a zobrazení trasy k danému cíli.
Tato aplikace slouží pro běžné použití laiky, kteří se mohou registrovat a využívat tak základní funkce aplikace např. možnost vyhledávat místa ve společnostech a využít vyhledávaní na mapě. Je zde také jiné rozhraní pro uživatele, kteří budou tuto aplikaci používat pro správu aplikace. Administrátorům tak umožní v aplikaci přidávat nové body zájmu, upravovat je, nebo je smazat z databáze. Další funkcí je aktivace a deaktivace uživatelských účtů.

1 Analýza problému

Základní funkcí aplikace je zobrazit na mapě zájmové body z vlastních databází. Tuto problematiku jsem řešil jelikož není mobilní aplikace, která by řešila problémy při hledání učeben na škole. Aplikace se tedy zabývá tvořením trasy k vybranému zájmovému bodu. Tato aplikace byla navržena tak, aby jí mohla využívat nejen jedna škola.
Prvním krokem bylo tedy nutno vytvořit nějaké rozdělení uložených společností, které by aplikaci chtěli využívat. Dále bylo nutno oddělit od sebe typy uživatelů. Prvním typem byl obyčejný uživatel, který by aplikaci využil pro hledání míst a dalším typem uživatele je administrátor. Tomu bylo nutno přidat funkce na přidávání a editaci míst, aktivování nově registrovaných uživatelů.
Dalším krokem bylo vybrání SDK pro zobrazení map, které popisuji v následující podkapitole. Poté bylo nutno vyčíst z dokumentace map funkce, které dané SDK nabízí.
Výsledkem má být aplikace s možností zobrazení bodů z vlastní databáze, tak i z databáze Google. Dále by aplikace měla obsahovat možnost vytvoření trasy k vyhledávaným bodům. Další funkcí aplikace mělo být vyhledávání zájmových bodů v okolí zařízení.

1.1 Zvolení SDK pro zobrazení map

První problém, který byl nutný vyřešit, bylo jakou mapu si zvolit. Prvním nápadem bylo využít mapy od společnosti Seznam a to mapy.cz. Hlavní důvod byl ten, že API od mapy.cz je zcela zdarma. Dalším důvodem, proč zvolit tyto mapy, byla přehlednost map pro Českou republiku. Jediným nedostatkem těchto map je neveřejné API mapy.cz pro operační systém Android, ve kterém je aplikace napsána. Řešení tohoto problému bylo zvolení map od společnosti Google. Tyto mapy jsou kompatibilní jak s webovým rozhraním, tak s mobilní aplikací. Nevýhodou těchto map je zpoplatnění pro komerční práce, která je přibližně od 20000 načtení map. Google mapy nabízí více SDK, které jsou spojeny s mapou samotnou, které jsem uvedl již dříve.

1.2 Registrování uživatelů a jejich přihlášení

Pro využití veškerých funkcí, které aplikace obsahuje, je nutná registrace a následné přihlášení uživatele. V aplikaci je to velice důležitý prvek, neboť můžeme rozeznat, zda-li se jedná o administrátora, nebo běžného uživatele. Dále můžeme zjistit k jaké společnosti daný uživatel má být přiřazen.
Při implementaci registrace, bylo nutno vyřešit, které parametry má daný uživatel o sobě zadat, aby ho mohl administrátor rozeznat a následně aktivovat. Přidání aktivace uživatele slouží jako ochrana společnosti před únikem informací dané společnosti. Tudíž není možno se registrovat k libovolné společnosti a rovnou se přihlásit, bez vědomí administrátora, pro danou společnost. Tato vlastnost je uložená v databázi pod atributem Approved, který má výchozí hodnotu nula tzn. deaktivován.

Při registraci a deaktivaci uživatelů je zadáváno heslo, zde musí být implementována hašovací funkce, aby hesla všech uživatelů nemohla být zneužita. Zde jsem použil Password hash.

1.3 GooglePlacesAPI a vydání nové aktualizace

Při řešení AutoCompleteSuggestion (dále jen ACS), které mělo vyřešit automatickou nabídku vyhledávání míst jsem se dostal do situace, která žádného programátora nepotěší a to vydaní nové aktualizace. Během ledna jsem vyřešil tento problém v tehdejší aktuální verzi *com.google.android.gms:play-services-places:16.0.0*, ta však 29. ledna 2019 přestala být aktuální a některé její funkce přestali fungovat. Musel jsem provést kompletní aktualizaci toho SDK jelikož, zbylé funkce, které se nezměnili by fungovali pouze do 29. července 2019. Poté se plně přechází na novou verzi *com.google.android.libraries.places:places:1.0.0*, která nahradí dosavadní SDK. Metoda ACS tak prošla zásadní změnou, tudíž adaptéry, které firma Google poskytovala již nefungovali. Tato metoda prošla velkou změnou a došlo k zjednodušení její implementace, ale tím, že jsem měl implementaci hotovou ve staré verzi, tato úprava trvala déle než se zdálo, z důvodu nepoškození zbylé části programu. Navíc tato aktualizace nebyla na internetu dlouho a sebemenší problém bylo velmi složité dohledat v dokumentacích. [7]

6. února vyšla také nová aktualizace od GoogleMapsAPI a to z předchozí verze *com.google.android.gms:play-services-maps:16.0.0* na novou verzi *com.google.android.gms:play-services-maps:16.1.0*, zde však došlo jen k přidáním nových funkcí a odstraněním problému s podporou Apache HTTP klienta, který byl ve starší verzi řešen přidáním tohoto pravidla do AndroidManifest.xml. S novou verzí se tato část kódu mohla smazat. [8]

2 Rozbor technologií

2.1 Androidu

Android je mobilní operační systém, založený na jádře softwaru Linux. Tento software je využíván v zařízeních např. chytré telefony, tablety, chytré televize aj.. Vývoj operačního systému vede firma Google od roku 2005, kdy si najala vývojáře na tento operační systém. Systém byl představen v roce 2007 a nyní je nejrozšířenějším operačním systémem na mobilních zařízeních na světě. Android umožňuje vývojářům psát v jazyce Java a používat knihovny vytvořené společností Google. Android je volně šiřitelný operační systém. Firmy tak mohou systém upravovat podle jejich představ, avšak musí se držet pravidel, které udává licence Apache. Android má nespočet vývojářů, kteří píší aplikace pro tento operační systém po celém světě. V únoru roku 2012 bylo ke stažení 450,000 aplikací a počet jejich stažení přesahoval přes 10 miliard. Aplikace pokrývají celou škálu kategorií např. hry, zábava, finanční služby, zpravodajství atd.. Vývoj softwaru Android a Google Play obchodu jsou relativně otevřené a neomezené. To nabízí vývojářům i uživatelům větší svobodu a možnost výběru z velké škály aplikací. [1]

2.2 Verze Androidu

Android je pravidelně aktualizován od doby jeho vydání. Tyto aktualizace se soustředí na opravu zachycených chyb a na přidávání nových schopností operačního systému, pro zvýšení pohodlí daného uživatel či vývojáře. Obvykle je daná verze androidu pojmenována podle nějakého dezertu a počáteční písmena verze jsou seřazeny podle abecedy, tak jak byly vydány, jak si můžete všimnout na obrázku. [1]

Obrázek 1: Evoluce operačního systému android

Na obrázku není zahrnuta nejnovější verze androidu 9.0 s názvem Pie. O prvních dvou verzích se nemá smysl rozepisovat, jelikož nepřinesli žádnou zásadní novinku. Jednalo se spíše o opravy

chyb. V raných fázích androidu vycházeli aktualizace několikrát do roka. Od verze 4.1 JellyBean začali vydávat nové verze přibližně jednou za rok.

2.2.1 Cupcake

Cupcake 1.5byl vydán na konci dubna roku 2009 a úroveň API byla 2. Tato verze přidávala do systému věci, které dnes bereme jako samozřejmost každého mobilního telefonu. Jednalo se o podporu widgetu, prohlížeče internetu, navigační aplikace, které mohli pracovat s mapou a fungovat jako navigační zařízení. Obsah šel synchronizovat a aktualizovat přes internet bez nutnosti připojení k počítačovému systému.[2]

2.2.2 Donut

Donut 1.6 byl vydán čtyři měsíce později s úrovní API 3. V této verzi byla přidána funkce zachycení snímku obrazovky nebo-li screenshot. Dále bylo přidáno ovládaní pomocí hlasových příkazů všech dostupných funkcí z předchozí verze softwaru.[2]

2.2.3 Eclair

Eclair se objevil hned po měsíci od vydání verze Donut. Objevil se ve verzích 2.0 až 2.1. Úroveň API se dostala na verzi 7. Aktualizace obsahovala vylepšení uživatelského rozhraní. Dále zde byla zavedena podpora HTML5 a podpora Exchange ActiveSync verze 2.5. [1]

2.2.4 Froyo

Froyo představil nový JavaScript engine a to Chrome V8 a optimalizoval JIT funkce, díky kterým se zvýšila výpočetní rychlost. Byla přidána nová funkce a to vytvoření Wi-Fi hotspotu a přidání podpory pro Adobe Flash. Tato aktualizace vyšla v květnu roku 2010. Poslední verze Froya byla 2.2.3 a úroveň API byla 8. [2]

2.2.5 GingerBread

GingerBread zavedlo podporu pro nový způsob přenosu dat NFC(Near Field Communication). Verze 2.3 - 2.3.7 byla vydána v prosinci roku 2010. Úroveň API se dostala na verzi 10. Aktualizace obsahovala také snížení spotřeby energie při běhu softwaru, dále přidání funkcí kopírování a vložení na klávesnici. [1, 2]

2.2.6 Honeycomb

Verze 3.0 - 3.2.6 z února 2011, přinesla řadu změn. API úroveň vzrostla na 13. Vydaní tabletů si vyžádalo podporu větší obrazovky, tudíž se zavádí mnoho nových funkcí pro uživatelské rozhraní. Podporuje více-jádrové procesory a hardwarové akcelerace pro grafické karty. Po vydání Honeycomb SDK, bylo představeno první zařízení s touto verzí a to Motorola Xoom tablet.[1]

2.2.7 Ice Cream Sandwich

Poslední aktualizací v roce 2011 byla verze 4.0 - 4.0.4 s úrovní API 15. Tato aktualizace je založená na změnách, které byly v předchozí verzi 3.2.6 pouze pro tablet zařízení. Verze byla kompatibilní se zařízeními jak s malým, tak velkým rozlišením. Byla posílena schopnost multitaskingu operačního systému, ve které zůstavají otevřené aplikace a je možné mezi nimi přepínat. Byl vylepšen vzhled oznámení odstraněním vyskakovacích oken v horní části obrazovky. Pokud je zařízení zamknuté do této verze nemohl uživatel nic dělat. Nyní může uživatel reagovat na příchozí hovory, nebo například přepínat hudbu bez nutnosti odemknutí zařízení. V reakci na zájmy uživatelů, byla vylepšeny funkce fotoaparátu, které zahrnovaly zvýšenou rychlost snímaní, panorama pro fotografie, nepřetržité ostření, nulovou závěrku. [2]

2.2.8 JellyBean

V červenci roku 2012 vyšla verze 4.1 - 4.3 s úrovní API 18. Tato verze vylepšila všechny předchůdce, ne však z hlediska rychlosti a výkonu, ale v uživatelském rozhraní. Pro plynulý obraz měla verze již obnovovací frekvenci na šedesáti snímcích za sekundu. Dále přišla podpora pro pět jazyků, které jdou zprava doleva, bezpečnostní opatření, správa digitálních práv, nízkoenergetická technologie Bluetooth pro audio, videohovory nebo vzdálený přístup. Nevýhodou této verze operačního systém byla možnost získávání dat o poloze, dokonce i při vypnuté funkci WiFi. [2]

2.2.9 KitKat

Koncem roku 2013 přesněji koncem října vyšla verze 4.4 - 4.4.4. Zde byla úroveň API 20. Tato verze byla přizpůsobena na mnohem větší zařízení než předchozí verze operačního systému. Bylo nutno navýšit minimální požadavky pro zařízení, které mohou používat tuto verzi operačního systému, a to RAM paměť musela mít minimálně 512MB. Dále se tato verze zaměřovala jak na uživatelské rozhraní, tak na rychlost a výkon. To bylo dosaženo díky vylepšené zpětné vazbě senzorů a použití tří-jádrových procesorů. Zlepšilo se využívání baterie, což zvýšilo její životnost. Došlo ke zvýšení rozlišení a bylo upraveno schéma pro uživatelské rozhraní. V této verzi se objevila technologie "edge to edge", která umožnila uživatelům nejen vzdálený přístup k zařízení, ale také využívat cloudový tisk, správu zdrojů a mnoho dalších. [1, 2]

2.2.10 Lollipop

Android Lollipop verze 5.0 - 5.1.1 s úrovní API 22 byl vydán v listopadu roku 2014. V této verzi vylepšili dosavadní 3D zobrazení. Nyní lze objekty prohlížet v reálném, čase se stíny při pohybu objektu. Přechod vizuálních prvků z jednoho stavu do druhého je nyní více plynulý a bezproblémový. S novým designem jsou vlákna nastavena pro plynulé vykreslování animací, i když dochází ke zpožďování hlavního vlákna aplikace. To, ale neohrožuje výkon systému, který

běží na novém Ahead of time JIT. Android Run time kódy jsou vytvořeny od nuly, aby vyhovovali potřebám, které měly 64-bit MIPS a ARM architektury. Lollipop splnil efektivně snížení odpadu a dokázal udržet aplikaci v souladu s jejím výkonem. Byly zavedeny nové senzory, jedním ze senzorů byl senzor na měření tepové frekvence. Další senzory sloužily pro detekci interakcí např. interakce s pohybem, švihnutím, sevřením atd. a senzor pro snímaní náklonu. Nově přidána rozhraní API zlepšila výkon baterie, dostupnost a zobrazení webů, snímání obrazovky a funkce fotoaparátu. [2]

2.2.11 Marshmallow

Verze 6.0 - 6.0.1 Marshmallow vyšel v říjnu roku 2015, API úroveň byla 23. V této aktualizaci se objevila velká novinka v oblasti mobilních telefonů a to přidání snímače na detekci otisků prstů, díky nimž můžeme např. odemknout zařízení, zvednout příchozí hovor, pořizovat fotografie nebo začít nahrávat video a mnoho dalších. V předchozích verzích se při instalaci nových aplikací museli udělit práva, ale později už se nedali změnit. Tato verze umožňuje uživateli změnit práva pro danou aplikaci i po nainstalování. Pro rychlejší nabíjení byl zaveden nový kabel typu C. Režim doze umožňuje operačnímu systému ušetřit více energie a téměř zdvojnásobit výdrž baterie. Upozornění na telefonu a kamera nyní můžou být použity při zamknuté obrazovce [2]

2.2.12 Nougat

Andorid Nougat verze 7.0 - 7.1.2 s úrovni API 25 vyšel v srpnu roku 2016. Bylo přidáno 72 nových emotikonů, celkem je jich přes 1500. Klávesnici si můžeme nastavit pro jazyk kterým píšeme, aby nápověda odpovídala danému jazyku. Byla přidána funkce rozdělení obrazovky a možnost pustit dvě různé aplikace najednou a jednoduše mezi nimi přecházet. Režim doze byl vylepšen. Nyní funguje lépe, i když telefon neleží na stejném místě. Režim se aktivuje, pokud telefon nosíme v kapse nebo kabelce. Dále bylo přidána možnost úpravy rychlého nastavení. Upozornění, které jsou od stejné aplikace se nyní zabalují do balíčků a nezabírají tolik prostoru v oznámeních. Dále byla přidána možnost rychlé odpovědi přímo u upozornění např. odpovědět na přijatou zprávu bez otevření aplikace. Přidána funkce Data Saver, která pomáhá šetřit mobilní data. [10]

2.2.13 Oreo

O rok později v roce 2017 vyšla verze 8.0 - 8.1 Nougat s API úrovní 27. V této verzi byly opět přidány nové možnosti k notifikacím. Byly přidány notifikační kanály, které si uživatelé mohou různě upravovat např. nastavovat jim důležitost, barvu, zvuk, zobrazení v módu nerušit aj.. Do kanálu lze přidat více zobrazení najednou. Další změnou byla notifikační tečka u aplikací, která značí upozornění hned při ikoně aplikace a pokud ikonu podržíme, otevře se aplikace na daném upozornění. Byly přidány adaptivní ikony, nyní dokáží měnit tvar podle typu zařízení. Vývojáři si tam můžou nastavit vlastní vzhledy ikon pro různé druhy telefonů. Byla opět zvýšena výdrž baterie, jelikož Google omezil aktivity běžící v pozadí zařízení. Nově zde byla uvedena funkce

obraz v obraze. Nyní si můžeme zmenšit např. video na YouTube do malého okénka na obrazovce a můžeme dále používat jiné aplikace, bez přerušení přehrávání daného videa. [11, 12]

2.2.14 Pie

Poslední vydanou aktualizací androidu, byla verze 9.0 Pie s API úrovní 28, která vyšla v srpnu roku 2018. Menu pro usnadnění přístupu je ve verzi 9.0 o mnoho jednodušší pro uživatele se sníženou pohyblivostí např. pořízení snímků obrazovky a práce s jednou rukou. Byla vylepšena funkce Select to speak. Nyní můžeme pomocí hlasu vybrat napsaný text a obsah bude přečten nahlas. Pomocí fotoaparátu s nastavením v režimu text můžeme vyfotit napsaný text, který se zvýrazní a bude přečten. Byl přidán zvukový zesilovač, který usnadňuje pochopení konverzací na hlučnějších místech např. restaurace, koncert atd.. S touto verzí přišla možnost připojit až pět Bluetooth zařízení a jednoduše mezi nimi přecházet. Příchozí hovor se zobrazí na všech připojených zařízeních. Opět byla snížena spotřeba baterie. Byla vylepšena funkce adaptivního jasu. Telefon se naučí, jak správně nastavit jas v daných světelných prostředích a časem to začne automaticky nastavovat. V této verzi bylo přidáno mnoho dalších menších funkcí a vylepšení těch stávajících. [13]

2.3 Android Studio

Android studio je vývojové prostředí od společnosti Google. Slouží k vývoji aplikací pro operační systém android. Lze programovat ve třech různých jazycích Java, Kotlin a C/C++. Toto prostředí obsahuje Visualní Layout Editor, k jednoduššímu vytváření grafické stránky aplikace. Součástí programu je mobilní emulátor, pomocí kterého můžeme provádět testování a debugování vývojáři napsané aplikace. Obsahuje flexibilní sestavovací systém. Je poháněn systémem Gradle, pomocí kterého si můžeme nastavit více variací pro různé druhy zařízení. Vývojové prostředí Android Studio je zcela zdarma.

2.4 GoogleAPI

Společnost Google nabízí řadu svých SDK a API, které si mohou vývojáři zapůjčit na jejich stránkách. Tyto služby jsou zdarma pro aplikace, které nejsou komerční a slouží pro vlastní potřebu. Popíši zde tři API, které jsem použil pro tuto práci. Všechny API, které Google nabízí, mají na internetu svou dokumentaci, která pomáhá vývojářům zorientovat se ve všech funkcích co dané API nabízí. Všechny API požadují API klíč, který je unikátní. Pomocí API klíče, lze zjistit, jak je aplikace s tímto klíčem využívána a Google si podle toho kontroluje, zda-li vývojář už musí za služby zaplatit.

2.4.1 Maps SDK pro Android

Pokud si přidáme toto SDK do našeho programu, program dokáže přidat mapy založené na datech od Google Maps. Toto API automaticky zvládá připojení k Google Maps serverům,

stahování dat, zobrazování dat na displeji a rozpoznává gesta pro pohyb mapy. Dále díky API můžeme implementovat funkce pro přidání značek, polygonů a k přechodu pohledu na jinou oblast. Tyto objekty poskytují další informace pro umístění map a umožňují interakci uživatele s mapou. Rozhraní dovoluje přidat následující grafické objekty do mapy[4]:

- Marker - ikona, která je ukotvena na specifických souřadnicích na mapě, dokáže vypsat informace o místě po kliknutí na Marker
- Polylines - sada křivek, která se používá u vykreslování tras apod.
- Polygons - používá se k zvýraznění oblasti mezi třemi a více body
- Ground Overlays - jedná se o bitmapovou grafiku ukotvenou na dané pozici na mapě
- Tile Overlays - sady obrázků, které se zobrazují v horní části mapy

2.4.2 Places SDK pro Android

Places SDK nám dovolí využívat geografická místa a body zájmů v okolí našeho zařízení. Places API vrací informace o místech na základě HTTP požadavků. K dispozici jsou následující požadavky na vyhledávaní míst.

- Place search - vyhledávaní míst, vrací seznam míst na základě umístění uživatele nebo vyhledávacího okna
- Place details - podrobnosti o místech, vrátí podrobnější informace o konkrétním místě, včetně uživatelských recenzí
- Place photos - fotografie umístění, poskytují přístup k milionům uložených fotografií v databázi Google Places
- Place autocomplete - automatické vyplnění, automaticky nabídne jméno, či adresu místa ve vyhledávacím okénku, vrací místa podle toho, co píše uživatel
- Query autocomplete - automaticky dokončený výraz, poskytuje predikce dotazů pro textově geografická vyhledávání

Ke každé z těchto služeb se přistupuje pomocí požadavku HTTP a vrací hodnotu JSON nebo XML. Všechny požadavky musí požadovat protokol https:// a musí obsahovat API klíč. [6]

2.4.3 Directions SDK pro Android

Google Directions je služba, která slouží pro výpočet vzdáleností mezi dvěma nebo více body na mapě pomocí HTTP požadavků. API vrací nejefektivnější cestu při počítání trasy, kde cestovní čas je primárním faktorem. Dále však bere v potaz vzdálenost, počet odboček a spoustu dalších faktorů, které mohou ovlivnit efektivnost trasy. API nabízí tyto funkce[3]:

- Vyhledávání tras pro základní typy dopravy např. cestování autem, chůzí, jízdou na kole, nebo MHD dopravou
- Vrací více možných tras podle různých faktorů
- Můžeme určit místa mezi začátkem trasy a jejím koncem. Trasa se přizpůsobí tak, aby vyhovovala všem podmínkám, které byly zadány.

2.5 MySQL databáze

MySQL je nejoblíbenější open source databáze na světě. Díky osvědčenému výkonu, spolehlivosti a snadnému použití se MySQL stala hlavní volbou pro webové aplikace. Databáze je využívána weby jako např. Facebook, Twitter a mnoho dalších. Tato databáze byla vytvořena ve Švédsku a společně s jazykem PHP tvoří oblíbenou dvojici pro vývojáře. Dále se je nejčastěji kombinovaná s Linuxovým operačním systémem a Apache webovým serverem. MySQL nabízí několik typů engine. Mezi dva nejoblíbenější enginy patří InnoDB a MyISAM. Tyto enginy mají podstatné rozdíly mezi sebou. Engine MyISAM je optimalizovaný pro čtení velkého množství dat. Důvodem je použití full-textového indexování. Tento engine však není bezchybný. MyISAM nepodporuje transakce a kontrolu cizích klíčů, oproti databázi InnoDB, která toto podporuje. [14]

2.5.1 Co je to databáze?

Databáze je samostatná aplikace, která umožňuje ukládat kolekce dat. Každá databáze má jedno nebo více odlišných rozhraní API pro vytváření, přístup, správu, vyhledávání a replikace dat, které jsou zde uloženy. Místo databáze, lze použít jiný typ ukládání dat, jako například ukládání do souborů, nebo ukládání do velkých hash tabulek. Tento způsob však není efektivní. Data se nedokáží tak rychle načítat a psát do souborů, jako právě do databáze. V současné době se využívají relační systémy pro správu databází, ukládaní a správu obrovského objemu dat. Nazýváme to relační databáze, protože všechny data jsou uložena v různých tabulkách a vztahy mezi nimi jsou vytvořeny pomocí primárních klíčů nebo ostatních, které jsou známe jako Foreign Keys. Relational DataBase Managment System (dále jen RDBMS) je software, který:

- Umožňuje implementovat databázi s tabulky, indexi a sloupci.
- Automaticky aktualizuje indexy.
- Zaručuje integritu mezi řádky různých tabulek.

- Zobrazuje dotaz SQL a kombinuje různé informace.

2.5.2 InnoDB

InnoDB je univerzální úložný engine, který vyvažuje vysoký výkon s vysokou spolehlivostí. V MySQL 8.0 je to výchozí úložný engine. Pokud se tedy nezadá jiný modul při vytvoření tabulky, je automaticky vytvořena tabulka InnoDB. Výhodami tohoto enginu jsou:

- Pokud dojde na serveru k hardwarovým nebo softwarovým chybám, bez ohledu na to co databáze vykonávala, nemusíme řešit po opravě chyb nic zvláštního. InnoDB obsahuje funkci InnoDB crash recovery, která automaticky dokončí všechny změny, které byly provedeny před časem havárie a zruší všechny změny, které byly v procesu a nebyly potvrzeny.
- Při rozdělování souvislých dat do různých tabulek můžeme nastavit cizí klíč, který vynucuje referenční integritu.
- Pokud se data na disku nebo v paměti poškodí, mechanismus kontrolního součtů nás upozorní před použitím neplatných dat.
- Můžeme komprimovat tabulky a přidružené indexy.
- Výhody výkonu nejsou omezeny na obří tabulky s dlouhodobými dotazy. Když jsou tázané stejné řádky pořád dokola, funkce Adaptive Hash Index přebírá tyto dotazy ještě rychleji, než z hašovací tabulky.

Právě v tomto enginu je napsaná i databáze, kterou využívá má aplikace. Zvolil jsem tento engine z důvodu možnosti přiřazení cizích klíčů a zlepšení integrity databáze. [15]

2.6 Java

Programovací jazyk Java je vysokoúrovňový jazyk, který je objektově orientovaný, multiplatformní a nezávislý na architektuře. Tento jazyk představila firma Sun Mircosystems v roce 1995. V roce 2010 tuto firmu odkoupila společnost Oracle, která pokračuje ve vyvíjení tohoto programovacího jazyka. Výhodou tohoto jazyka je jeho jednoduchost v syntaxích oproti jazyku C a C++. Jazyk neobsahuje ukazatele, bez znaménkové datové typy aj..
Platoforma Java obsahuje dva komponenty. Jedním z nich je Java Virtual Machine a druhý Java Application Programming interface (API). Program se píše v čistých textových souborech s příponou .java, které jsou kompilovány do souborů .class. Soubor neobsahuje kód speciálně pro váš procesor, jako u jiných jazyků, ale obsahuje tzv. bytecode, který je určený pro Java Virtual Machine. [19]
Java je kompatibilní s mnoha operačními systémy např. Linux, Windows, MacOS a další. Společnost Google využila tento jazyk pro tvorbu operačního systému Android, přestože Android stojí na Linuxovém základu a hlavním programovacím jazykem byl jazyk C. [18]

2.7 PHP

PHP celým názvem HypertextPreprocessor je skriptovací programovací jazyk. Je to alternativa pro technologii Active Server Page od společnosti Microsoft. PHP je serverový, multiplatformní, HTML vestavěný skriptovací jazyk. PHP skripty se spouští na webovém serveru, nikoliv v prohlížeči na vašem počítači. Skripty mohou běžet na různých operačních systémech a webových serverech. PHP je skvělý nástroj pro tvorbu dynamických a interaktivních webových stránek a také pro komunikaci s databází. PHP je velmi rozšířená, efektivní a hlavně bezplatná technologie. Jedná se o open source technologii, která běží na webovém serveru Apache. V PHP jsou napsány i velké internetové stránky např. Wikipedia nebo Facebook. Výhodou PHP je nativní podpora s velkým množstvím databázových systémů, obsahuje podrobnou dokumentaci. [16, 17]

3 Implementace

Pro implementaci jsem zvolil vývojové prostředí Android Studio společnosti google, které je založeno na IntelliJ IDEA. Hlavním programovacím jazykem je Java a pro práci s grafickou části aplikace je použit jazyk XML. Pro propojení s databází jsou použil skriptovací jazyk PHP.

3.1 MySQL databáze

Do databáze ukládám informace o společnostech, uživatelích, bodech a cestách.

3.1.1 Tabulka Corp

V tabulce Corp jsou uloženy jednotlivé společnosti, které jsou přiřazeny do systému. Tabulka obsahuje sloupce:

- CorpID - jedná se o ID dané společnosti datového typu int(11), pomocí kterého se v programu orientuji jaké body mají být zobrazeny a kteří uživatelé k ní mohou přistupovat
- Name - oficiální název společnosti, datový typ varchar(255)
- City - město, ve kterém se společnost nachází, datový typ varchar(255)
- Longitude - zeměpisná délka, odkazuje na hlavní místo ve společnosti(ředitelství, rektorát), datový typ float(10,6)
- Latitude - zeměpisná šířka, odkazuje na hlavní místo ve společnosti(ředitelství, rektorát), datový typ float(10,6)

3.1.2 Tabulka users

V této tabulce jsou uloženi všichni registrovaní uživatelé. Obsahuje tyto parametry:

- user_id - ID uživatele, které je přiřazeno automaticky, datový typ int(5)
- FirstName, LastName - další dva sloupce obsahují celé jméno uživatele, jméno slouží pro orientaci, o kterého uživatele se jedná, datový typ varchar(255)
- City - město, ve kterém uživatel bydlí, tento parametr pomáhá v situaci, kdyby došlo ke shodě jmen, datový typ varchar(255)
- Login - unikátní přihlašovací jméno uživatele, datový typ varchar(255)
- Pass - zašifrované heslo uživatele, datový typ varchar(255)
- Mail - email uživatele, datový typ varchar(255)
- Admin - zde je uložena informace, zda-li je uživatel jako Admin, datový typ tinyint(1)

- Approved - zde je uložena informace, zda-li je účet od uživatele potvrzen, datový typ tinyint(1)
- CorpID - odpovídá ID společnosti, pro kterou se uživatel zaregistroval, datový typ int(11)

3.1.3 Tabulka Mark

V tabulce Mark jsou uloženy informace o přidaných bodech společnosti, které lze zobrazit na mapě. Obsahuje tyto parametry:

- id - ID markeru, přidáváno automatickou inkrementací, slouží pro orientaci v databázi, datový typ int(11)
- Longitude - zeměpisná délka přidaného místa, datový typ float(10,6)
- Latitude - zeměpisná šířka přidaného místa, datový typ float(10,6)
- Nazev - název místa na mapě, datový typ varchar(255)
- CorpID - ID společnosti, pro kterou byl bod přidán, datový typ int(11)
- Info - informace o daném místu, které se zobrazí u ukazatele na mapě, datový typ varchar(255)

3.1.4 Přístup k databázi pomocí PHP

Pro přístup k MySQL databázi jsem zvolil připojení pomocí skriptovacího jazyku PHP. Přímé připojení k databázi nebylo nejlepší volbou, jelikož JDBC Connector, pomocí kterého šlo toto propojení uskutečnit není na aktuální verzích androidu podporované. Tudíž jsem zvolil přístup pomocí do databáze přes PHP soubory, které jsou umístěny na webhostingu, který je zařízený od společnosti Endora.
Připojení k databázi mi v programu obstarává třída Connection, ve které vytvářím HTTP připojení k URL adrese uloženého PHP souboru. Ve všech případech využívám metodu POST. Po úspěšném připojení k PHP souboru si data, která chci odeslat do databáze, zabalím do JSONObject, které následně parsuji do tvaru, který je požadován pro úspěšné spojení př. &key=value. Zabalení do požadovaného tvaru mi obstarávají třídy začínající na DataPacked. Každý DataPacked je samostatný, důvodem je lepší orientace v kódu a následné aktualizace programu. Poté si vyžádám pomocí funkce getResponseCode() zprávu zdali proběhlo připojení v pořádku nebo došlo k nějaké chybě. Pokud je ResponseCode roven číslu 200 jedná se o standardní odpověď pro úspěšný HTTP požadavek. Odpověď z PHP získávám ve tvaru JSONObject př. {key=value}. Objekt parsuji a informace si uložím.
Všechny toto úkony jsou implementovány v třídách začínající názvem Sender. Třídá Sender dědí ze třídy AsyncTask, ve které je nutno implementovat metodu doInBackground(). Dále jsem

z třídy AsyncTask implementoval další dvě metody onPreExecute() a onPostExecute(). V metodě onPreExecute() se implementuje ProgresDialog, který se zobrazí dokud aktivita, která se vykonává v metodě doInBackground() neskončí. V metodě onPostExecute() zpracovávám informace obdržené z PHP souboru a pomocí třídy Toast, zobrazuji text uživateli jak daná činnost dopadla.
Tímto způsobem řeším veškeré propojení databáze s aplikací. Tento způsob je univerzální v počtu zadaných parametrů a nezatěžuje program neustálým udržováním spojení.

3.2 Java

Jako hlavní programovací jazyk jsem zvolil Javu. Java je objektově orientovaný jazyk. Tento jazyk lze použít na různých platformách tzn. multiplatformní.

3.2.1 AndroidManifest

Každá android aplikace musí obsahovat souboru AndroidManifest.xml. Tento soubor popisuje jak se má daná aplikace sestavit, jaká práva má požadovat po uživateli při prvním spuštění aplikace a využívaní GooglePlay služeb jako např. Mapy, využití GooglePlay přihlášení aj..
V mé aplikaci jsem potřeboval právo k přístupu internetu zjišťování polohy a zápisu do externí paměti.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Výpis 1: Vyžádaní práv telefonu v AndroidManifestu

3.3 Registrace a přihlášení

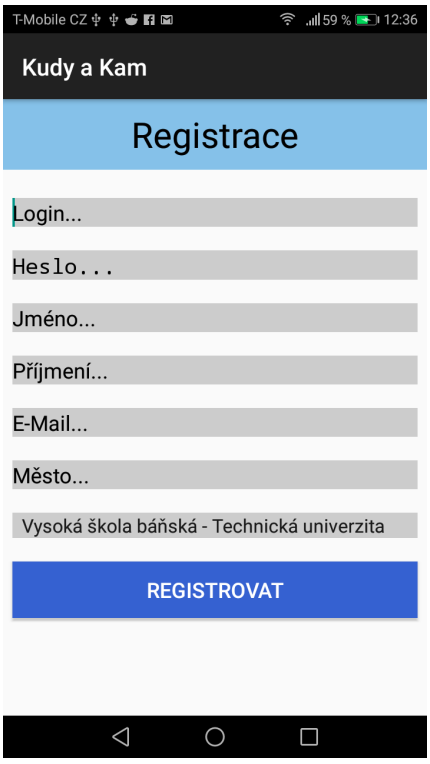
3.3.1 Registrace

V aktivitě RegistrationActivity jsou implementovány pouze EditTexty, Spinner a tlačítko. Všechny EditTexty je pro úspěšné registrování nutno vyplnit. Při registraci je nutno zadat tyto parametry:

- Login - uživatelské jméno, pomocí kterého se budou uživatele přihlašovat
- Heslo - neveřejné heslo, které si uživatel sám zvolí
- Jméno a příjmení - tyto informace jsou nutné pro pozdější orientaci, pro administrátory, kteří musí účet aktivovat, tudíž musí znát jejich jméno a příjmení

- E-mail - hlavním důvodem emailu je resetování zapomenutých hesel po registraci. Dalším důvodem je možnost kontaktování uživatele e-mailem. Je nutné, aby e-mail byl zadán ve správném tvaru, je zde přidáno ověření a to pomocí atributu, který lze nastavit v XML souboru u Edittextu a to parametr android:inputType. Ten je zvolen na typ textEmailAddress.
- Město - kdyby došlo ke shodě jmen a administrátor znal město jeho bydliště, mohl by se podle toho orientovat
- Spinner - je to kontejner ve kterém jsou uloženy přiřazené společnosti, pro které je aplikace navržena. Jejich počet lze snadno navýšit. Pro demonstraci jsou společnosti dvě

Pokud uživatel vše vyplní a stiskne tlačítko registrova, spustí se třída SenderReg.java. V této třídě se spustí metoda doInBackground() ,v níž se nejprve vytvoří připojení s PHP souborem mobile_registration.php. Poté se veškeré zadané informace zabalí a pošlou se POST metodou do PHP souboru. PHP soubor pošle na vyžádaní response, který obsahuje zprávu, jak registrace dopadla. Pokud by bylo zvoleno stejné uživatelské jméno jaké již v databázi bylo použito, aplikace to uživateli oznámí Toast zprávou zobrazenou na displeji. Taktéž se stane pokud se registrace podaří nebo nebudou vyplněny všechny položky.



Obrázek 2: Aktivita registrace

3.3.2 Přihlášení

V aktivitě LoginActivity jsem implementoval dva Edittexty, které složí pro přihlašovací jméno uživatele a heslo. Zadávané heslo se skrývá do hvězdiček, toto je nastaveno přes atribut u EditTextu inputType. Pokud uživatel vše vyplní a stiskne tlačítko přihlásit se, vyvolá se metoda SenderLogin, přes kterou se vytvoří připojení s PHP souborem mobile_check.php. PHP mi vrací JSONObject, který obsahuje veškeré informace o uživateli. Podle zprávy, kterou tento objekt obsahuje, nastavím následnou aktivitu a uložím si do proměnné ID společnosti, ke které je přiřazen.

Obrázek 3: Aktivita přihlášení

3.4 GoogleAPIs

Abych mohl používat jakékoliv API od společnosti Google, je nutné si nejdříve aktivovat na jejich internetových stránkách a poté přidat APIkey do naší aplikace. V této aplikaci využívám tři různé API, které jsou spojené s GoogleMaps. A to GoogleMapsAPI, GooglePlacesAPI a GoogleDirections.

3.4.1 GoogleMapsAPI

Pro zobrazení map od společnosti Google je nutné si vytvořit aktivitu MapActivity, která je v prostředí Android studia už připravena. Základní metody této aktivity jsou onCreate() a on-

MapReady()
V metodě onCreate inicializuji veškeré přidané widgety a přiřazuji jim jejich ID, které mají v XML souboru. Metoda onMapReady slouží k zjištění, zda-li bylo přiděleno oprávnění k získání informací o poloze, pokud nebylo, tak si ho aplikace dotazem vyžádá viz. metoda getLocation-Permition. Dále tato metoda nastavuje další parametry mapy např. zapnutí zjišťování aktuální polohy, nastavení adaptérů aj..

```
String[] permissions = {FINE_LOC, COARSE_LOC};

if(ContextCompat.checkSelfPermission(this.getApplicationContext(), FINE_LOC) ==
    PackageManager.PERMISSION_GRANTED){
if (ContextCompat.checkSelfPermission(this.getApplicationContext(), COARSE_LOC)
    == PackageManager.PERMISSION_GRANTED) {
permisionGranted = true;
}else{
ActivityCompat.requestPermissions(this,
permissions,
123);
}
}else{
ActivityCompat.requestPermissions(this,
permissions,
123);
}
}
```

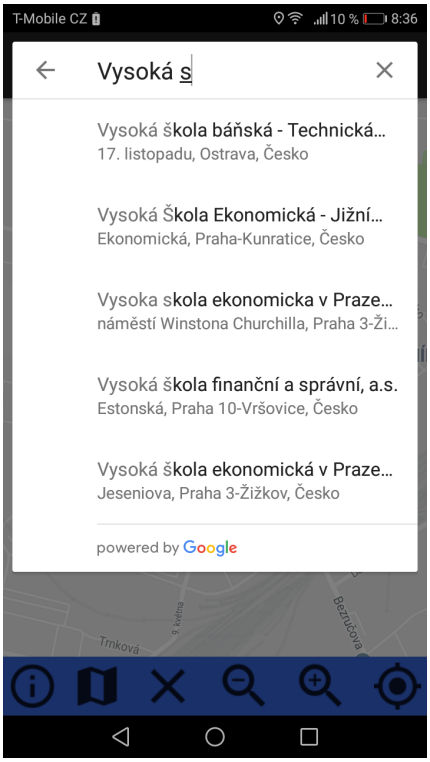
Výpis 2: Metoda getLocationPermition

Dále bylo nutné vyřešit inicializaci všech posluchačů, kteří sledují jakékoliv změny pokud se klikne na nějakou ikonu, pokud se začne vyhledávat text. Všechny tyto funkce jsou zahrnuty v metodě init(), která je taky volaná ve funkci onMapReady().

V aplikaci mám dvě aktivity, které využívají GoogleMapsAPI a to třídu MapsActivity.class a CorpMaps.class. Zvolil jsem rozdělení těchto dvou částí do vlastních aktivit z důvodů využití jiného stylu vyhledávání bodů. V třídě MapsActivity.class jsem využil AutoComplete suggestion od společnosti Google, aby bylo možno vyhledat bez jakýchkoliv potíží libovolné místo v České republice. V třídě CorpMaps.class jsem musel řešit jiný problém a to zobrazení vlastních míst z databáze. Pro tento případ bylo nutné si implementovat vlastní AutoComplete suggestion adaptér. Dalším důvodem proč jsou tyto mapy oddělené je, že mapy v aplikaci může využít i neregistrovaný uživatel.

3.4.2 Autocomplete suggestion

Jak jsem již zmiňoval v kapitole 3.1, tak s novou aktualizací se výrazně funkce zjednodušila. Nyní stačilo inicializovat třídu Places. Poté jsme vytvořili AutocompleteSupportFragment, kde se zobrazují možné volby míst. Nastavil jsem zde omezení, aby to nabízelo nabídku jen z České Republiky, jelikož při testovaní vyhledávání jsme museli napsat skoro celé hledané místo než nám nabídlo to, co jsme hledali. Dále jsem uvedl o jaké parametry mám zájem, aby jsem je při zvolení daného místa mohl zobrazit do vlastního informačního pole, které se objeví po kliknutí na značku zobrazenou na mapě, nebo po stisknutí ikony informace. Tyto data jsem si uložil do proměnné typu PlaceInfo a poté zavolal metodu moveCamera s tímto parametrem.[9]



Obrázek 4: AutocopleteSuggestion

3.4.3 Metoda moveCamera

Tato metoda je přetěžována, a to z důvodu jaké informace o daném místě mám. Pokud znám jen název místa, použiji jinou metodu než pokud znám veškeré informace o daném místě. V této metodě vytvářím Snippet z informací, které jsem si již dříve uložil. Snippet je jeden z možného nastavení v MarkerObtions objektu. Poté jen přidám značku do mapy pomocí následujícího řádku. Přidávám zde ukázku, jak se mění pozice kamery na daný bod. Parametry metody CameraUpdateFactory jsou poslány při zavolání metody moveCamera() jako její parametry.

```
//přidání značky do mapy
mMarker = mMap.addMarker(options);

//pohyb kamery na novou lokaci
mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng,zoom));
```

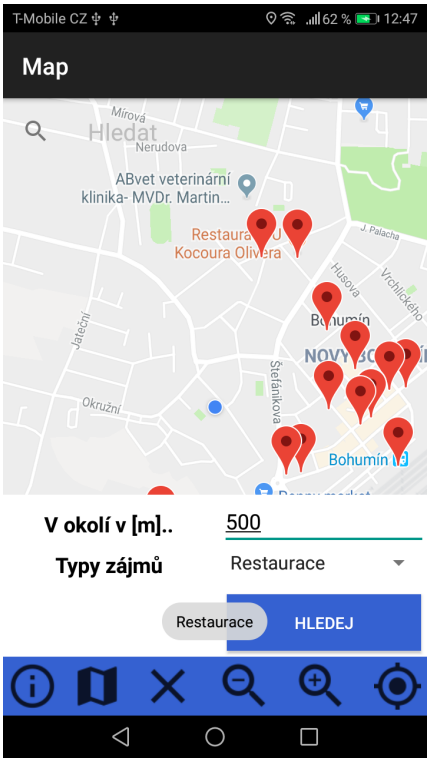
Výpis 3: Metoda moveCamera

3.4.4 Metoda NearbyPlaces

Google Places API jsem využil nejen u AutocompleteSuggestion, ale také u získávání informací zájmových bodů, které se vyskytují v okolí zařízení, ve zvoleném rádiu. Základem této metody je zjistit jaké zájmové body jsou v našem okolí. Uživatel si tak zvolí poloměr od své polohy a vybere typ zájmového bodu (restaurace, bankomaty, školy aj.). Tyto informace si převedu do požadovaného tvaru od API viz. https://maps.googleapis.com/maps/api/place/nearbysearch/output?parameters. Parametry které jsou nutné nastavit jsou:

- key - zde je nutno vložit autorizační klíč tzn. APIkey. Tento klíč nám vygeneroval Google, při aktivování jejich API.
- location - je udáván ve tvaru location=latitude,longitude. Prvním parametrem je zeměpisná šířka a druhým zeměpisná výška. Tyto parametry nastavují odkud se má dané místo vyhledávat. Já to nastavuji na aktuální polohu zařízení.
- radius - poloměr vyhledávaní, udávaný v metrech.
- type - typ zvoleného zájmového bodu z možného výběru. Google nabízí přes 50 různých možností. V mé aplikaci jsem vybral pro ukázku čtyři, ale navýšit jejich počet není už žádný problém.

Po získání informací z URL, které jsou ve formátu JSONObject, je nutné, dané informace parsovat. Data jsou uložena do hašovací tabulky, kde si uložíme potřebné informace o místu. Z hašovacích tabulek vytvoříme list, pro uschovaní všech nalezených bodů. Ve třídě GetNearbyPlacesData, která dědí ze třídy AsyncTask, se v metodě showNearbyPlaces přiřazují informace k ukazateli. Je zde Java funkce for, aby to proběhlo u všech bodů, které jsou uloženy v daném listu. Po zobrazení bodů na mapě, lze na každý ukazatel kliknout a po kliknutí se zobrazí informační okénko, ve kterém jsou napsány základní informace o daném místu. Kdybychom chtěli vytvořit trasu k danému místu, musíme kliknout na zobrazené informační okno. O této metodě se rozepíši v následující kapitole.



Obrázek 5: Metoda NearbyPlaces

3.4.5 Vytváření trasy k bodům na mapě

Možnost vytvoření trasy nabízím po kliknutí na informační okénko vyhledaného místa. Po kliknutí se zobrazí AlertDialog, který se ptá jakým způsobem se chceme dopravit na dané místo. Tyto možnosti jsou uloženy ve Spinneru, tudíž uživatel musí před vytvořením trasy vybrat z možností.
Pro výpočet trasy využívám Google Directions API, se kterým komunikuji pomocí URL adresy. Do adresy musím přidat následující parametry:

- origin - tento parametr obsahuje aktuální polohu uživatele, tudíž zeměpisnou šírku a délku
- destination - tento parametr uvádí polohu naší destinace
- mode - zvolený způsob dopravy na dané místo
- key - API klíč

Po vytvoření URL adresy, se spustí metoda getDirectionsData.execute(). V této metodě stáhnu příslušná data z URL adresy, a ty poté parsuji ve třídě DataParser. Tato třída obsahuje více metod, které mi z těchto dat dovolí získat cestu k danému místu. Tuto cestu vrátím zpět do třídy GetDirectionsData, kde se vykreslí nejefektivnější trasa k zvolenému místu do mapy.

3.5 Přidávání, úprava a mazaní bodů z databáze

Bylo nutné přidat tyto funkce programu do aplikace, aby mohl administrátor spravovat místa dané společnosti přímo z mobilní aplikace. Tyto funkce tedy vidí jen uživatelé, kteří mají přidané administrátorské právo.

3.5.1 Přidávaní nové lokace do databáze

V této třídě jsou v XML souboru vytvořeny čtyři EditText okna a jedno tlačítko. Pro úspěšné přidání je nutno vyplnit všechny informace, pokud je nevyplníme vyskočí informační hláška. Nutné parametry pro přidání bodu jsou:

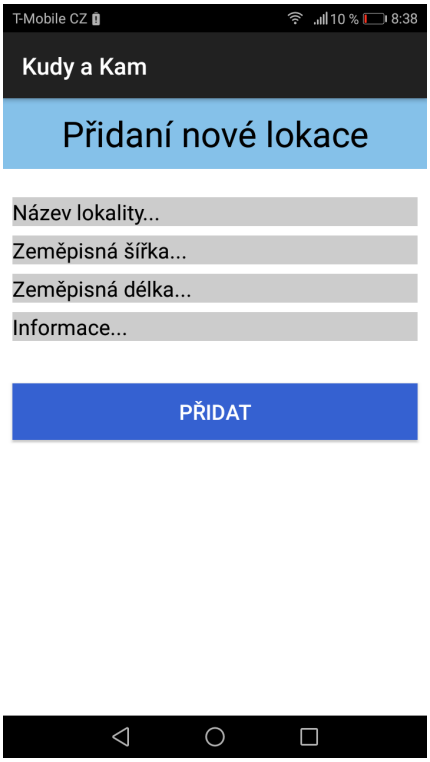
- Název - název daného místa, kterým se pak bude volat pro zobrazení
- latitude - zeměpisnou šířku daného místa
- longitude - zeměpisnou délku daného místa
- Info - informace o daném místě. U naší školy se například přidává informace název budovy plus podlaží, ve kterém se daná učebna nachází
- CorpID - tento parametr je neviditelný, jelikož nesmí být upraven, jedná se o ID dané společnosti, pro kterou je dané místo tvořeno. Tuto informaci si předávám od přihlášení uživatele při změně aktivity pomocí funkce putExtra() a v následující aktivitě pomocí getStringExtra uložím do proměnné

Po stisknutí tlačítka se vyvolá funkce s.execute(); viz. následující ukázka.

```
button.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
DataPackedNewDestination.SenderNewDestination s~= new DataPackedNewDestination.
    SenderNewDestination(NewDestination.this, urlAdress, corpID, editName,
    editLat, editLong, editInfo);
s.execute();
}
});
```

Výpis 4: SenderNewDestination vyvolání metody

V této třídě dojde k zabalení veškerých dat do tvaru potřebného k navázaní POST přenosu do souboru PHP. Po úspěšném poslání, obdržím z PHP souboru JSONObject, kde zjistím jak to dopadlo. Následně to vypíši na obrazovku pomocí Toast.makeText, abych informoval uživatele, zda-li bylo místo úspěšně přidáno, nebo se vyskytla chyba.



Obrázek 6: Aktivita pro přidání nového místa do databáze

3.5.2 Úprava, smazání lokace z databáze

Abych jsem mohl upravovat lokaci z databáze a nemusel si pamatovat jakou jsem chtěl upravit, bylo nutné stáhnout z databáze všechny uložené lokace a vložit je do listu. Pro zobrazení všech parametrů daného místa bylo nutno vytvořit si vlastní ListViewAdapter a vlastní XML soubor listview_item.xml.

Obrázek 7: Aktivita pro výpis všech uložených míst v databázi

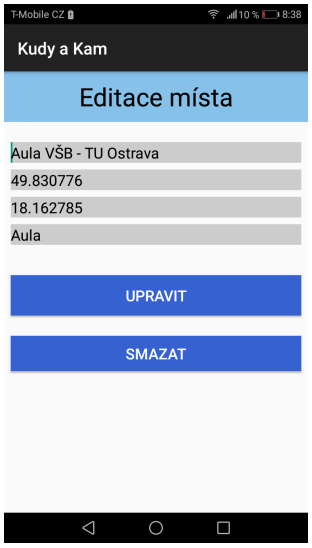
V XML souboru jsem implementoval, jak by měla vypadat jedna položka listu. Chtěl jsem, aby obsahovala veškeré údaje o daném místě, tedy její název, zeměpisnou šířku, zeměpisnou délku a informace. Tyto údaje byly v programu přiřazeny pomocí setText do daného TextView. Poté bylo nutné implementovat třídu ListViewAdapter. ListViewAdapter dědí ze třídy BaseAdapter a využil jsem některé z jejich abstraktních metod, např. getCount(), getItem() aj.. Vytvořil jsem si zde třídu ViewHolder, která mi ukládala informace daných TextView. Poté jsem implementoval metodu getView(), ve které jsem přiřadil vytvořený XML soubor, uvedený výše v textu. Přiřadil jsem zde ID od TextView proměnám ve ViewHolderu, abych jim mohl přiřadit daný text. Je zde implementován taky OnClickListener, který po kliknutí veškeré informace bodu uloží do intentu a pošle do nové aktivity UpdateDestinationEdit. Informace jsem stahoval z databáze opět pomocí PHP souboru, který mi vrátil výsledek ve tvaru JSONObject, který jsem si parsoval viz. následující výňatek z metody. V proměnné response je uložen String ve formátu JSONObjectu. Dále jsem si vytvořil třídu UpdateDestGetSet, která obsahuje metody Get a Set pro všechny proměnné a metodu toString(). Tuto třídu jsem potřeboval, abych mohl vytvořit list, který bude obsahovat všechny informace.

```
try {
JSONObject jsonResponse = new JSONObject(response);
JSONArray jsonArray = jsonResponse.getJSONArray("results");
for (int i = 0; i < jsonArray.length(); i++) {
JSONObject r = jsonArray.getJSONObject(i);
this.ListData.add(new UpdateDestGetSet(r.getString("Nazev"), r.getString("
    CorpID"), r.getString("Latitude"),
r.getString("Longitude"), r.getString("Info"), r.getString("id")));
}

}catch (JSONException e){
Log.e("TAG", "jsonResponse: JSONException" + e.getMessage());
}
```

Výpis 5: Parsování získaných míst z databáze

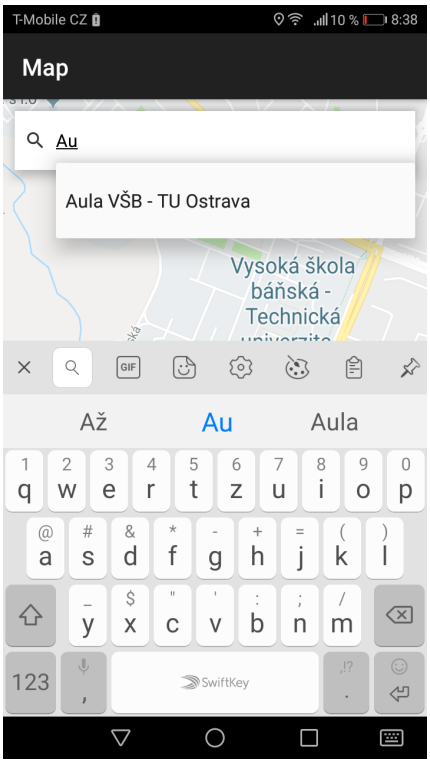
Po kliknutí na daný prvek v listu se spustí nová aktivita, ve které jsou čtyři EditTexty už předvyplněné informacemi daného místa a dvě tlačítka Upravit a Smazat. Uživatel bude moci upravit jakýkoliv parametr a odeslat aktualizovanou verzi do databáze. Po odeslání se aktivita zavře a zobrazí se opět daný list s výpisem. Pokud stiskne tlačítko smazat, vyskočí na ně upozorňovací okénko, které se znovu zeptá, zda-li dané místo má být nadobro odstraněno z databáze. Pokud zvolí ano odešle se požadavek do PHP souboru a toto místo bude smazáno.



Obrázek 8: Editace daného místa

3.6 Vyhledávání míst z vlastní databáze

Pro vyhledávání míst z vlastní databáze jsem musel implementovat vlastní Autocomplete adaptér, který mi dokázal získat veškeré informace o daném místě. Vyhledávání funguje na dotazech na PHP soubor mobile_autocomplete.php uložený na webhostingu. Toto PHP mi vytahuje místa z databáze, které začínají na stejná písmena jaká byla poslána do databáze. Pokud nastane shoda, uložím si dané místo do fronty, kterou vracím jako JSONObject do programu. V programu tento objekt parsuji a výsledné informace ukládám do listu, který je typu SuggestGetSet. Tuto třídu jsem si vytvořil, abych mohl nastavit a zjistit získané informace z databáze. Třída SuggestionAdapter dědí ze třídy ArrayAdapter typu String. V metodě getFilter() zobrazuji veškeré místa, která mi databáze poslala zpět. Pokud kliknu na vyhledávané místo, objeví se na mapě značka na souřadnicích získaných z databáze. Po kliknutí na značku vidíme název místa a příslušné informace o místě. Po kliknutí na informační okénko, můžeme naplánovat trasu k místu, jak jsem již rozepsal v kapitole o vytváření trasy k místu.



Obrázek 9: Vyhledávání míst z vlastní databáze

3.7 Aktivace nových uživatelů

Implementoval jsem bezpečností funkci této aplikace, z důvodu některých útoků na aplikaci např. registrování do nějaké firmy za účelem zjištění firemních informací aj..
Tato aktivita obsahuje dva listy, ve kterých jsou uloženi aktivovaní a neaktivovaní uživatelé.

Pro získání všech uživatelů z databáze, vytvářím Http připojení a pomocí PHP souboru si všechny uživatele zabalím do JSONObject a v aplikaci parsuji ve třídě ParseActivateUser. Bylo nutné vytvořit nový ListView adaptér, ve kterém se uživatelé přiřazují do správného listu. Podle atributu Approved tak zjistím, zda-li je účet aktivní nebo nikoli.
Po kliknutí na vybranou položku v listu vyskočí AlertDialog, který se zeptá, zda-li chceme daný účet aktivovat. Po zvolení tlačítka ano, se spustí metoda SenderActivateUser, která pošle do PHP souboru změnu u atributu Approved. Tím se daný účet aktivuje, pokud byl neaktivní, nebo naopak. Aktivita obsahuje tlačítko obnovení, díky kterému se aktualizují uživatelé, u kterých došlo ke změně.

Obrázek 10: Aktivace uživatelů

3.8 XML layout

Grafická část aplikace se v programu Android studio řeší XML soubory, které si můžeme libovolně utvořit. V mé aplikaci jsem využil LinearLayout u většiny aktivit. LinearLayout nám umožní prvky ukládat v jednom směru, v mém případě ve směru vertikálním. Hlavními prvky layoutu, které jsem použil, jsou TextView, EditText, Spinner, Button. Pro určitá zobrazení jsem si potřeboval implementovat specifické layouty:

- Layout pro info window - tento layout obsahuje dva TextView, jeden slouží pro zobrazení názvu vyhledávaného místa, druhý slouží pro výpis informací o místě, obvykle zde ukládám vytvořený snippet, který tyto informace obsahuje.

- Dialog spinner - slouží k zobrazení Spinneru, v metodě onInfoWindowClick, kde přiřazuji možný výběr dopravy v AlertDialogu
- ListView item - v programu jsou vytvořeny dva, jeden pro list, který zobrazuje vytvořené místa z databáze a zobrazuje u nich jejich název, zeměpisnou šířku a délku a informace. Druhý se používá u zobrazení listu uživatelů, ten zobrazuje jméno, příjmení, login a město.

Závěr

Cílem práce bylo vytvoření mobilní aplikace pro prezentaci zájmových bodů z vlastní databáze.

V bakalářské práci jsem vytvořil funkční aplikaci, která pracuje s databází uloženou na webovém hostingu pomocí PHP souborů. Aplikace obsahuje funkce jako přihlášení a registrace nových uživatelů. Pro administrátora jsou implementovány funkce pro práci s databází. Tyto funkce zahrnují přidání a úpravu míst v databázi. Pro zobrazení map jsem použil API od společnosti Google. Vytvořil jsem dvě odlišné aktivity pro zobrazení map, jedna slouží pro orientaci v rámci České republiky a druhá právě v rámci společností, která zobrazuje body nahrané v databázi.

Využít tuto aplikaci budou moci noví studenti této školy, po vložení všech míst a učeben školy do databáze, pro orientaci v kampusu školy.

Literatura

[1] KIRTHIKA, B., S. PRABHU a S. VISALAKSHI. Android Operating System: A Review [online]. 2015 [cit. 2019-04-07]. ISSN 2394-9333. Dostupné z: https://www.researchgate.net/publication/327387842_ANDROID_OPERATING_SYSTEM_A_RE

[2] HARIS, Muhammad, Basit JADOON a Farhan Hassan KHAN. Evolution of Android Operating System: A Review [online]. 2017 [cit. 2019-04-07]. Dostupné z: https://www.researchgate.net/publication/319617606_Evolution_of_Android_Operating_System_A

[3] Directions API. Google Developers [online]. Google, 2019 [cit. 2019-04-10]. Dostupné z: https://developers.google.com/maps/documentation/directions/intro

[4] Maps SDK for Android. Google Developers [online]. Google, 2018 [cit. 2019-04-09]. Dostupné z: https://developers.google.com/maps/documentation/android-sdk/intro

[5] Places SDK for Android. Google Developers [online]. Google, 2019 [cit. 2019-04-09]. Dostupné z: https://developers.google.com/places/android-sdk/intro

[6] Place API. Google Developers [online]. Google, 2019 [cit. 2019-04-10]. Dostupné z: https://developers.google.com/places/web-service/intro

[7] Migrating to the New Places SDK Client. Google Developers [online]. Google, 2019 [cit. 2019-03-31]. Dostupné https://developers.google.com/places/android-sdk/client-migration#place_picker_and_autocomplete_updates

[8] Release Notes Maps SDK for Android. Google Developers [online]. Google, 2019 [cit. 2019-03-31]. Dostupné z: https://developers.google.com/maps/documentation/android-sdk/releases#2019-02-06

[9] Places Autocomplete. Google Developers [online]. Google, 2019 [cit. 2019-03-31]. Dostupné z: https://developers.google.com/places/web-service/autocomplete

[10] Android Nougat. Android [online]. Google [cit. 2019-04-08]. Dostupné z: https://www.android.com/versions/nougat-7-0/

[11] Android - 8.0 Oreo. Android [online]. Google [cit. 2019-04-09]. Dostupné z: https://www.android.com/versions/oreo-8-0/

[12] TRLICA, David. Android 8 novinky: Shrnutí hlavních funkcí. Svět Androida [online]. 28.8.2017 [cit. 2019-04-09]. Dostupné z: https://www.svetandroida.cz/android-8-shrnuti-novinek/

[13] Android 9 Pie. Android [online]. Google, 2018 [cit. 2019-04-09]. Dostupné z: https://www.android.com/versions/pie-9-0/

[14] VERZOVÁNÍ DATABÁZE PŘI VÝVOJI APLIKACÍ [online]. Brno, 2017 [cit. 2019-04-09]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=159258. Bakalářská práce. Vysoké učení technické v Brně. Vedoucí práce RNDr. Marek Rychlý, Ph.D.

[15] Best Practices for InnoDB Tables. MySQL [online]. [cit. 2019-04-10]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/innodb-best-practices.html

[16] PHP. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-11]. Dostupné z: https://cs.wikipedia.org/wiki/PHP

[17] PHP Technology. Suncore microsystem [online]. [cit. 2019-04-11]. Dostupné z: http://www.suncoremicrosystem.com/welcome/technology/27

[18] Java (programovací jazyk). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-12]. Dostupné z: https://cs.wikipedia.org/wiki/Java_(programovac%C3%AD_jazyk)

[19] Java tutoriál - Technologie (1. díl). Programujte [online]. 2007 [cit. 2019-04-12]. Dostupné z: http://programujte.com/clanek/2007040702-java-tutorial-technologie-1-dil/